PASSED

Computer lab 1

Kevin Neville, Gustav Sternelöv, Vuong Tran

1 februari 2016

Assignment 1: Be careful with '=='

A pupil of a school is bad in arithmetic but good in programming. He writes a program to check if 1/3-1/4==1/12.

```
x1<-1/3
x2<-1/4
if (x1-x2==1/12){
print("Teacher said true")
} else{
print("Teacher lied")}
```

```
## [1] "Teacher lied"
```

1.1. Check the result of this program. Comment why this happened.

This happened because the floating points of the different values may not be the exact values. This causes problems with the round-off for the compared values. *x1* subtracted by *x2* should be the same as *1/12*, but since the exact values of *1/3* and/or *1/4* not is represented by *x1* and *x2* the subtraction of the values not becomes exactly *1/12*.

Example:

$$\frac{1}{3} - \frac{1}{4} = 0.3333333 - 0.25 = 0.0833333, and \frac{1}{12} = 0.08333333$$

The left side of the equation differ from the right side of the equation with a value of

$$0.00000003 = 0.3 * 10^-7$$

1.2. Specify how the program can be modified to give a correct result.

This can be solved in many ways. For example by pre determining the amount of mantissa digits to 8 in the if statement, which results in a comparsion by 8 decimal precision.

```
x1<-1/3
x2<-1/4
if (round(x1-x2,8)==round(1/12,8)){
  print("Teacher said true")
} else{
  print("Teacher lied")
}
```

```
## [1] "Teacher said true"
```

Assignment 2: Derivative

A widely known way to compute the derivative of function f(x) in point x is to use

$$f'^x = \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

2.1. Write your own function computing the derivative of function f(x)=x in this way. Take $\epsilon = 10^{-15}$.

```
# 2.1
derivative_function <- function(x, epsilon){
  print(((x + epsilon) - x) / epsilon)
}
```

2.2. Compute your derivative function at point x=100000.

```
# 2.2
derivative_function(x=100000, epsilon = 10^-15)
```

```
## [1] 0
```

2.3. What is the value you obtained? What is the real value of the derivative? Explain the reason behind the discovered difference

The obtained value is *0* but the real value of the derivative is *1*. The reason behind this error is connected to the different magnitudes. Here x is a very large value and *epsilon* is a very low value. Therefore the addition of *x+epsilon* practically becomes x which means that in the numerator *x-x* is the executed computation made in practice. The denominator is still equal to the value of ϵ, this is because it does not interfer with any magnitude problems. This leads to a division where the value of *x-x* is divided by *epsilon*, which of course becomes *0*.

A known formula for estimating the variance is

$$Var(x) = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right)$$

Assignment 3: Variance

3.1. Write your own function myvar estimating variance in this way.

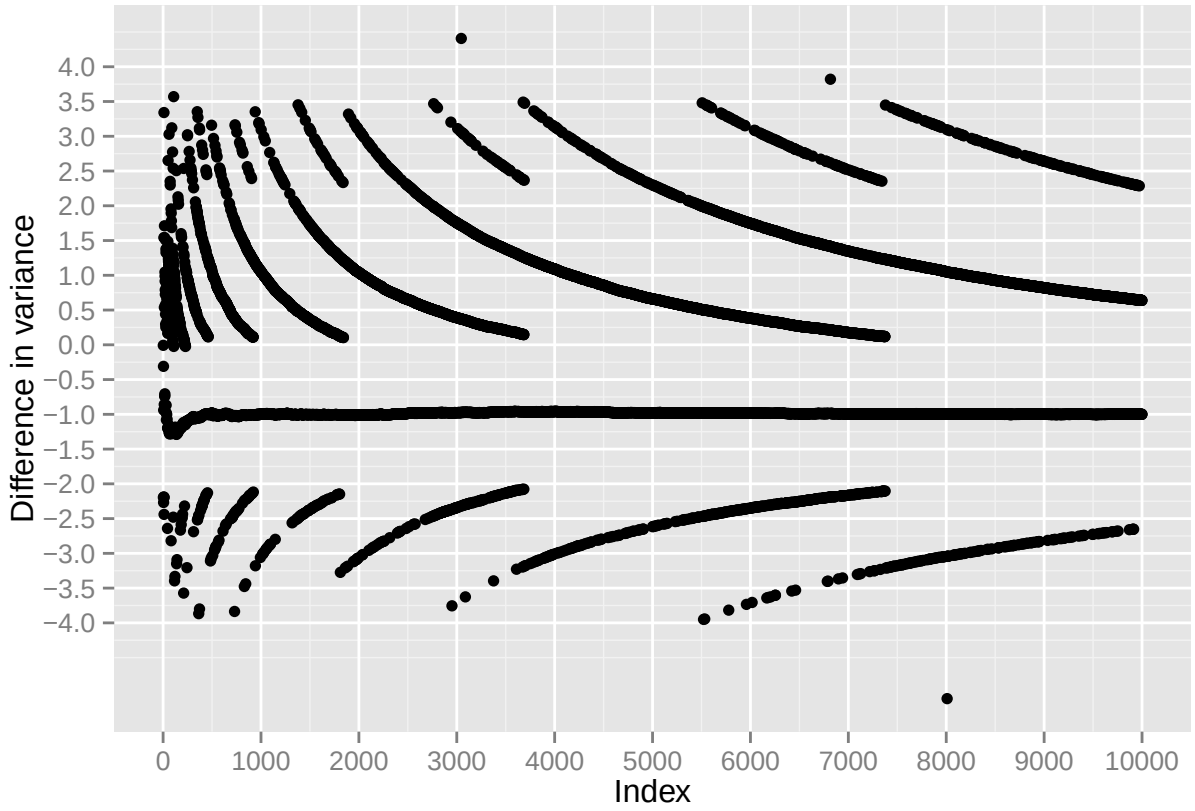
```
res <- NA
myvar <- function(x){

  for (i in 1:length(x)) {
    res[i] <- (1 / (length(x)-1)) * ((sum(x^2)) - (1/(length(x))) * (sum(x))^2)
  return(res)}
  }
```

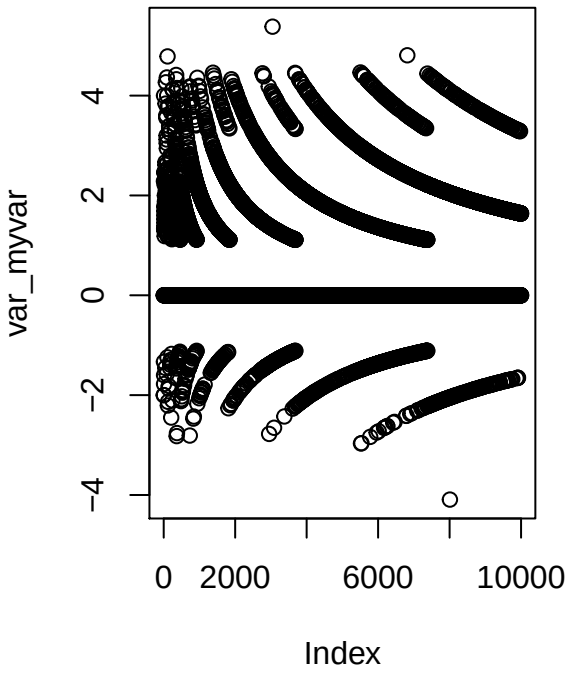
3.2. Generate vector x=(x1...x10000) with 10000 random numbers, normally distributed with mean 10^8 and variance 1.

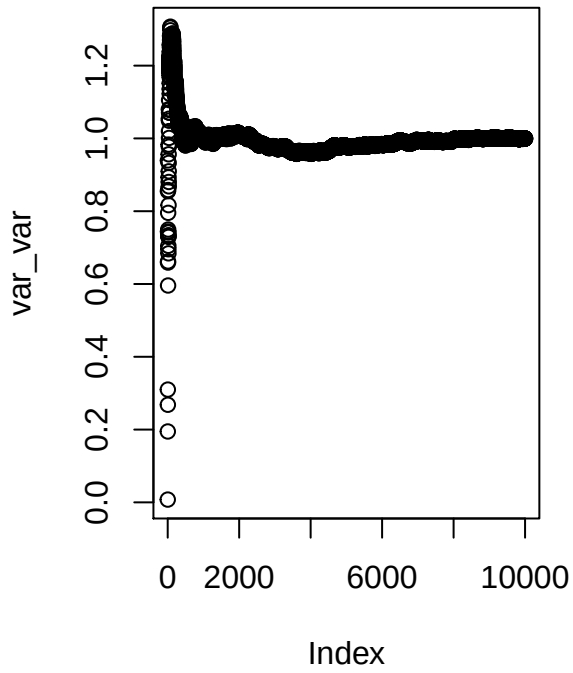
```
set.seed(12345)
x <- rnorm(10000, 10^8, 1)
```

3.3. For each subset $X_i = x_1...x_i$, i=1...10000 compute difference $Y_i = myvar(X_i) - var(X_i)$, where $var(X_i)$ is a standard variance estimation function in R. Plot the dependence Y_i on i. Draw necessary conclusions. How well does your function work? What is the reason behind such behavior?



According to the plot there is some difference in how the variance is calculated between the two functions. If the functions would be the same, the points in the plot should be focused around zero. But now they seem to be more focused around -1. The myvar function estimates of the varaince is not as precise as the ones from the var function. This leads to that the difference is not centred around zero.





Above is the estimated varainces of respective functions. We clearly see that the var function estimates improves with a larger sample.
The myvar functions does not provide as precise estimates, this is due to problems with summation of data with same sign and approximately same magnitude.

Assignment 4: Linear Algebra

The Excel file "tecator.xls" contains the results of a study aimed to investigate whether a near- infrared absorbance spectrum and the levels of moisture and fat can be used to predict the protein content of samples of meat. For each meat sample the data consists of a 100 channel spectrum of absorbance records and the levels of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer. The moisture, fat and protein are determined by analytic chemistry. The worksheet you need to use is "data". It contains data from 215 samples of finely chopped meat. The aim is to fit a linear regression model that could predict protein content.

4.1. Import the data set to R

4.2. Optimal regression coefficients can be found by solving a system of the type $A\beta = b$ where $A = X^T X$ and $b = X^T Y$. Compute A and b for the given data set.

```
# Estimating A
A <- t(X) %*% X

# Estimating b
b <- t(X) %*% Y
```

X is our design matrix which consists of all the predictor values and the first column contains ones to be used in estimating the intercept. The dimensions of the A matrix is 103 * 103. The Y vector is the response vector which contains the values of protein. The Y vector contains 103 values.

4.3. Try to solve $A\beta = b$ with default solver solve(). What kind of result did you get? How can this result be explained?

```
# Solve the equation
solve(a = A, b = b)
```

Applying the solve function on a=A and b=b result in an error message:
Error in solve.default(a = A, b = b) :
system is computationally singular: reciprocal condition number = 7.78804e-17
This is a consequnce of having a singular matrix, in other words the inverse of the matrix A can not be calculated.

4.4. Check the condition number of the matrix A and consider how it is related to your conclusion in step 3.

```
kappa(A)
```

```
## [1] 8.523517e+14
```

As noticed in 4.3 the linears system in not to be trusted. The value we obtain for the condition number of matrix A should then follow the same path. This is also the case since the obtain condtion number is very large. The condition number is a measure of how big the error could be in the worst case scenario.

4.5. Scale the data set and repeat steps 2-4. How the result has changed and why?

```
# Scale data
X_scaled <- scale(X[,-1], center = TRUE, scale = TRUE)
X_scaled <- cbind(intercept, X_scaled)

# Estimating A_scaled
A_scaled <- t(X_scaled) %*% X_scaled

# Estimating b
b_scaled <- t(X_scaled) %*% Y

# Solve the equation
head(solve(a = A_scaled, b = b_scaled))
```

```
##                [,1]
## intercept   17.68279
## Channel1  -333.77236
```

```
## Channel2  -667.26108
## Channel3  1140.32810
## Channel4  -391.27591
## Channel5  1247.15298
```

```
kappa(A_scaled)
```

```
## [1] 490471520661
```

When applying *solve* on the scaled matrices the *beta* coefficients for the respective x-variables are obtained by the ordinary least squares estimation. The problem that occured for the non-scaled data in *4.3* does not occur for the scaled data. Data has proved to be sensitive to scaling since there is a signficant difference between the magnitude of the values for the predictors, especially the variables *fat* and *moisture* has much higher values than the others.

The condition number for the scaled data is 4.9047152×10^{11}. The result has changed because of the scaling of data. The new condition number is lower and an interpretation of this could be that the scaled linear system performs better than the non-scaled. However, this does not necesserily need to be true. The lower condition number could be due to the scaling and moreover does not a lower condition number guarantee that the new system is better. A lower number is preferable but it does not completely assures that its better. For a linear system the condition provides a bound on the relative norms for the "correct" solution. Since it specifies the bound a lower value is preferable but a higher number does not have to mean that the linear system is worse.

Appendix

```
x1<-1/3
x2<-1/4
if (x1-x2==1/12){
print("Teacher said true")
} else{
print("Teacher lied")}
x1<-1/3
x2<-1/4
if (round(x1-x2,8)==round(1/12,8)){
  print("Teacher said true")
} else{
  print("Teacher lied")
}
# 2.1
derivative_function <- function(x, epsilon){
  print(((x + epsilon) - x) / epsilon)
}
# 2.2
derivative_function(x=100000, epsilon = 10^-15)
res <- NA
myvar <- function(x){

  for (i in 1:length(x)) {
    res[i] <- (1 / (length(x)-1)) * ((sum(x^2)) - (1/(length(x))) * (sum(x))^2)
  return(res)}
```

```
  }
set.seed(12345)
x <- rnorm(10000, 10^8, 1)
diff_var <- rep(NA, 10000)
for(i in 1:length(x)){
diff_var[i] <- myvar(x[1:i]) - var(x[1:i])
}

library(ggplot2)
gg_diff_var <- as.data.frame(cbind(diff_var, index=1:length(diff_var)))
qplot(y=diff_var,x=index, data=gg_diff_var, geom = "point",
      xlab="Index",ylab="Difference in variance") +
  scale_y_continuous(breaks=seq(-4, 4, 0.5)) +
  scale_x_continuous(breaks=seq(0, 10000, 1000))
par(mfrow=c(1,2))
var_var <- NA
for(i in 1:10000) {
  var_var[i] <- var(x[1:i])
}
var_myvar <- NA
for(i in 1:10000) {
  var_myvar[i] <- myvar(x[1:i])
}
plot(var_myvar)
plot(var_var)
par(mfrow=c(1,1))
library(gdata)
tecator <- read.xls("/Users/Kevin/Documents/Skola/732A38, Computational Statistics/Lab 1/tecator.xls",
                    sheet = 1)

# Extracting correct data and set as X and Y
X <- as.matrix(tecator[, -c(1, 103)])
Y <- as.matrix(tecator[, 103])
# Should an intercept be used?
 intercept <- rep(1, nrow(tecator))
 X <- cbind(intercept , X)
# Estimating A
A <- t(X) %*% X

# Estimating b
b <- t(X) %*% Y
## # Solve the equation
## solve(a = A, b = b)
kappa(A)
# Scale data
X_scaled <- scale(X[,-1], center = TRUE, scale = TRUE)
X_scaled <- cbind(intercept, X_scaled)

# Estimating A_scaled
A_scaled <- t(X_scaled) %*% X_scaled

# Estimating b
b_scaled <- t(X_scaled) %*% Y
```

```r
# Solve the equation
head(solve(a = A_scaled, b = b_scaled))

kappa(A_scaled)
## NA
```